The Center of Us

Jan de Leeuw - University of California Los Angeles

Started August 16, 2024; Version of August 18, 2024

Contents

1 The Data **3**

2 The Centroid **4**

3 The Weber Point **5**

4 The Minimax Point **7**

A Code **9**

A.1 center.R 9
A.2 weberIter.R 10
A.3 utility.R 11
A.4 convert.R 11
A.5 gcDistances.R 12
A.6 haversine.R 13
A.7 haversine.c 13

References **14**

Note: This is a working paper which will be expanded/updated frequently. All suggestions for improvement are welcome. All files are in the public domain. Attribution will be appreciated but is not required.

1 The Data

Since 2010 the De Leeuw family has owned, and lived in, houses in five different places. Their addresses are

- 4301 Irvon Trail, Frazier Park, CA 93225
- 11667 Steinhoff Road, Frazier Park, CA 93225
- 8 North Stafford Street, Portland, OR 97217
- 413 Shays Street, Amherst, MA 01002
- 235 Eastern Avenue, Springfiled, MA 01109

The coordinates of these five addresses, found using https://www.gps-coordinates.net, are

```
##             latitude   longitude
## irvon     34.8222245 -118.9575031
## steinhoff 34.8368579 -119.0761942
## stafford  45.5763561 -122.6670871
## shays     42.3417905  -72.5065297
## eastern   42.1049969  -72.5623351
```

The distances between these five addresses, in miles, are

```
##           irvon steinhoff stafford shays eastern
## irvon         0         7      768  2530    2528
## steinhoff     7         0      766  2535    2534
## stafford    768       766        0  2464    2468
## shays      2530      2535     2464     0      17
## eastern    2528      2534     2468    17       0
```

Note that all distances in this note are great circle (spherical) distances that take the curvature of the earth into account. Spherical distances were computed with the haversine function from the pracma package (Borchers (2023)). For completeness we also include in the appendix a version of haversine in C, with .C() interface glue, as well as an R function to compute the distances using the arcsine formula.

2 The Centroid

To compute the centroid (average) of the addresses we first convert to Cartesian coordinates, then take the average, and then convert back to spherical coordinates. This makes more sense than averaging latitudes and longitudes directly.

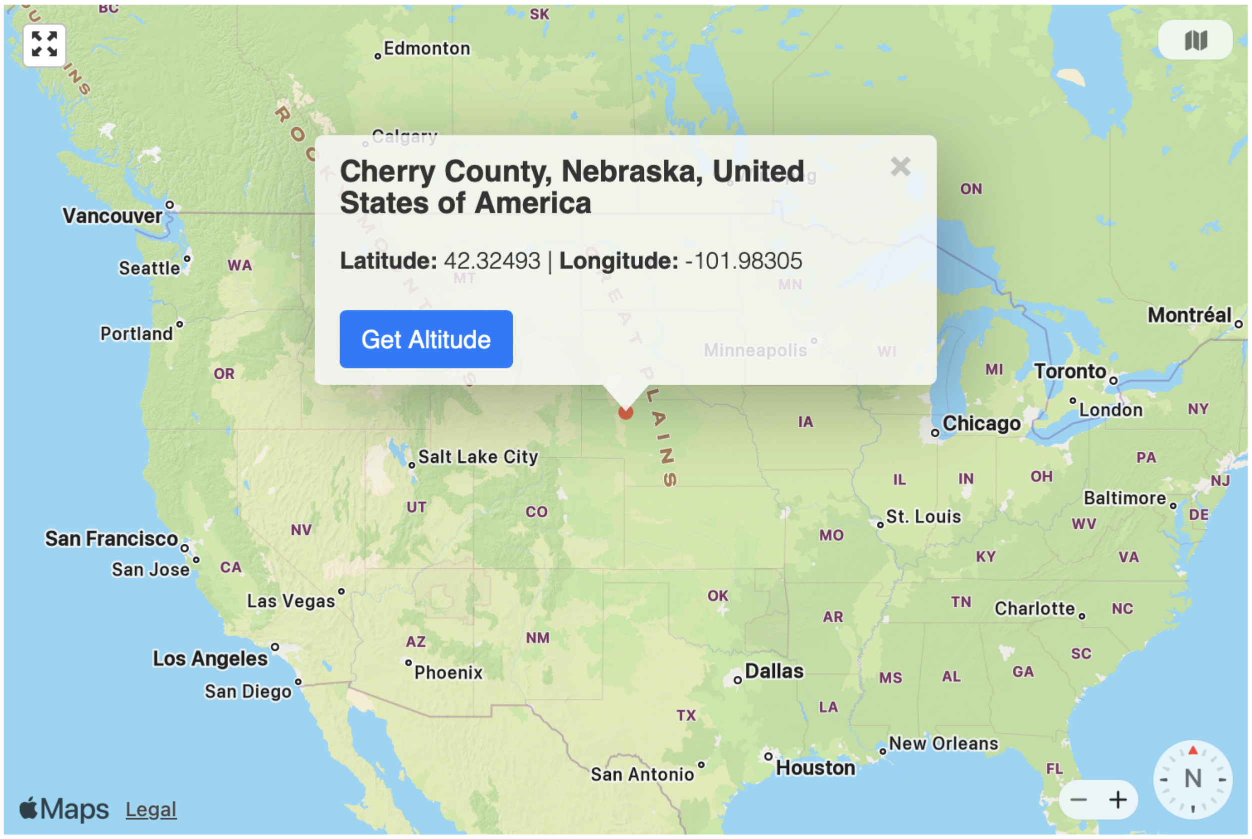
The latitude and longitude of the centroid point are

```
## 42.32493 -101.98305
```

This puts the centroid point in the high plains of Nebraska, just north of the Sand Hill dunes, close to Alliance, and on top of the Ogallala Aquifer.

The sum of the five spherical distances between the centroid point and the five addresses is 6152 miles. The five distances are

```
##           distance
## irvon         1050
## steinhoff     1055
## stafford      1049
## shays         1497
## eastern       1498
```


Cherry County, Nebraska, United States of America

Latitude: 42.32493 | Longitude: -101.98305

Figure 1: Centroid Point

3 The Weber Point

We next find the point for which the sum of the five spherical distances to the addresses is as small as possible. This is the Weber point. To compute the Weber point we use a spherical distance variation proposed by Katz and Cooper (1980) of the algorithm of Weiszfeld (1937) (translated in Weiszfeld and Plastria (2009)).

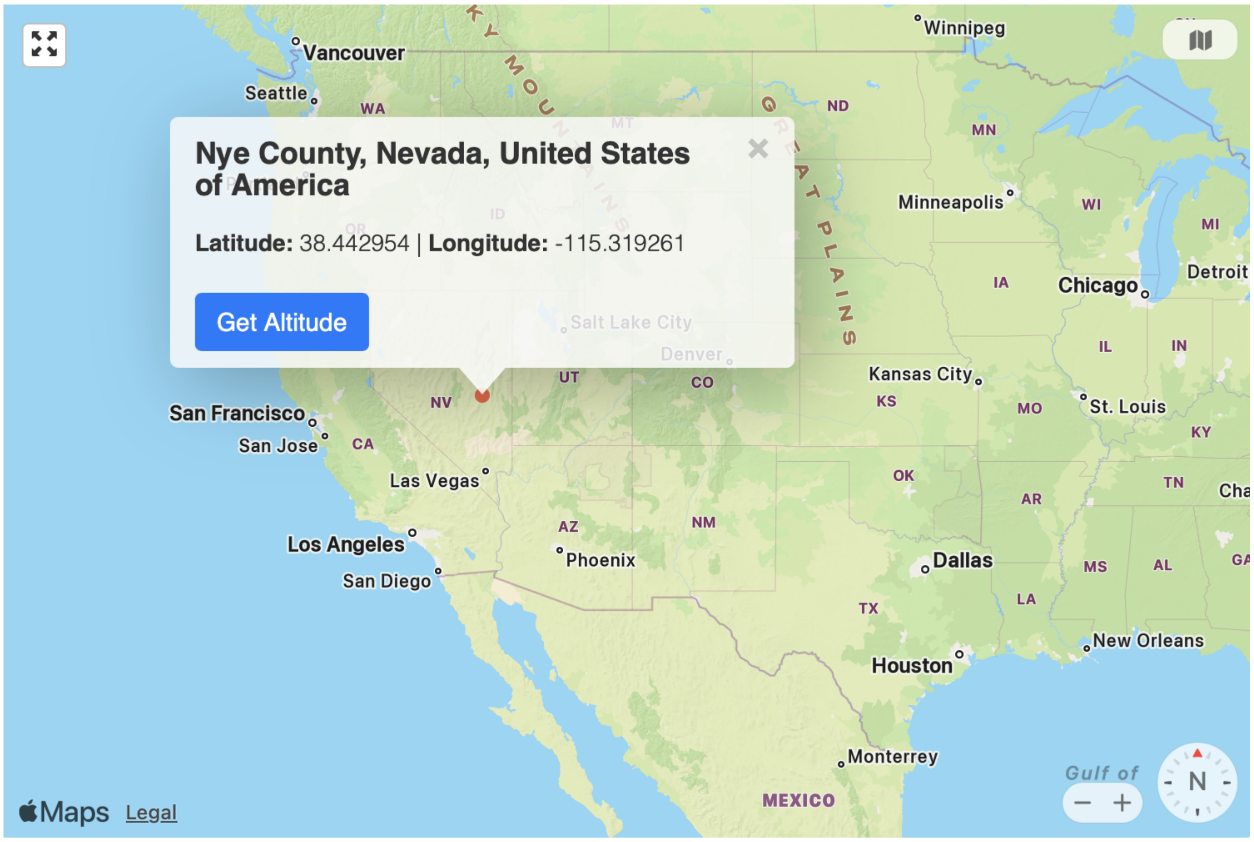
The Weber point has latitude and longitude

```
## [1]   38.44294887 -115.31926622
```

This puts it on the east slopes of the Grant Range mountains in east-central Nevada, close to Troy Peak, not exactly a central location between East Coast and West Coast.

The sum of the five spherical distances between the Weber point and the five addresses is 5756 miles. The five spherical distances are

```
##            distance
## irvon         321
## steinhoff     324
## stafford      620
## shays        2245
## eastern      2244
```


Nye County, Nevada, United States of America

Latitude: 38.442954 | **Longitude:** -115.319261

Figure 2: Weber Point

4 The Minimax Point

Instead of minimizing the sum of the spherical distances we can also minimize the largest of the five spherical distances. For this we use the optim function from the stats package in R (R Core Team (2024)).

The minimax point is located in north-east Nebraska, close to the Winnebago Reservation and to Norfolk, about 100 miles north-west of Omaha.

It has latitude and longitude

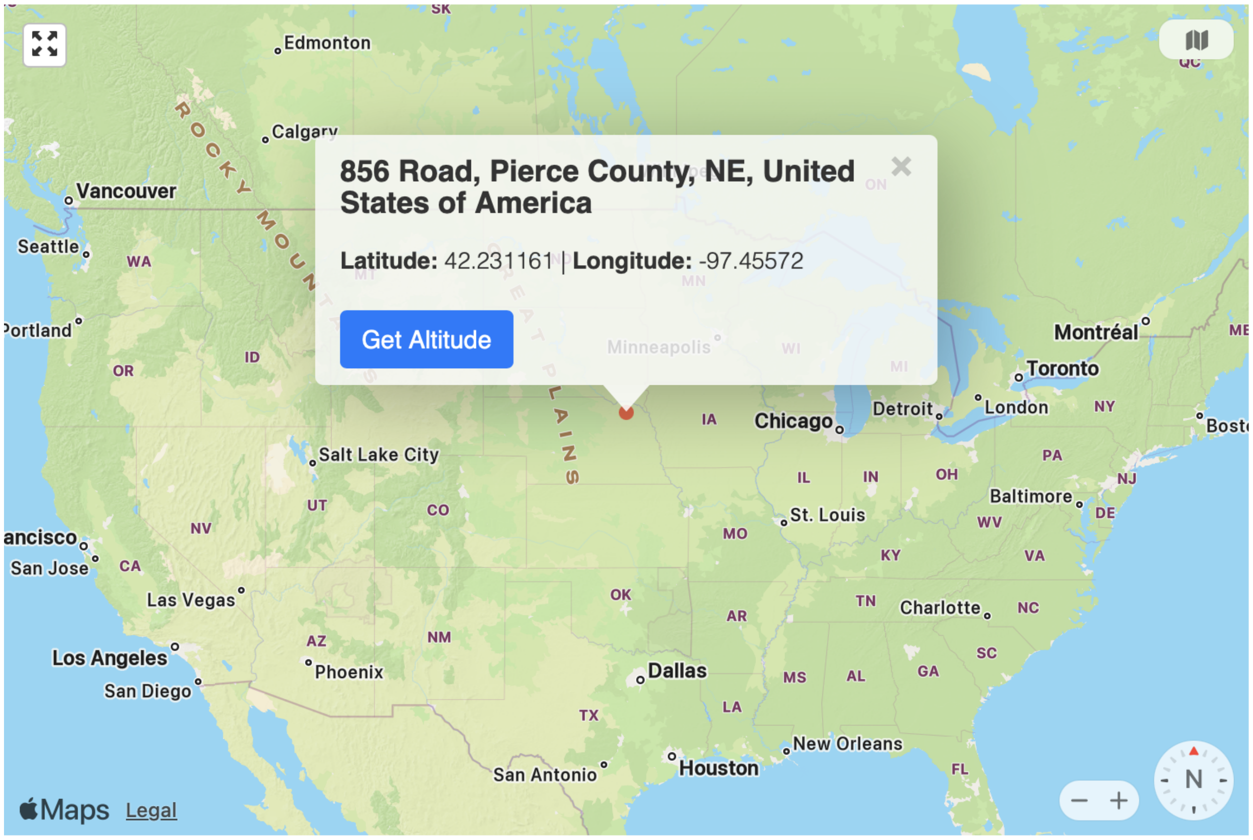
```
## [1]  42.23115887 -97.45571809
```

The largest of the five spherical distances between the minimax point and the five addresses is 1271 miles. The five distances are

```
##            distance
## irvon        1265
## steinhoff    1270
## stafford     1270
## shays        1270
## eastern      1270
```

and their sum is 6348 miles.

We see that all five addresses are practically on a circle with the minimax point in the center. This is not surprising. There are basically only three significantly different addresses: the pair {Irvon, Steinhoff}, the singleton Stafford, and the pair {Shays, Eastern}. There always is a circle through the three vertices of a triangle (the circumcircle of the triangle), and this is true both for flat and for spherical triangles.


856 Road, Pierce County, NE, United States of America

Latitude: 42.231161 | **Longitude:** -97.45572

Figure 3: Minimax Point

A Code

A.1 center.R

```r
centroid <- function(x) {
  xsum <- c(0, 0, 0)
  for (i in 1:nrow(x)) {
    y <- convertSphericalToCartesian(x[i, ])
    xsum <- xsum + y
  }
  xsum <- xsum / sqrt(sum(xsum ^ 2))
  return(convertCartesianToSpherical(xsum))
}

weber <- function(y, x) {
  dsum <- 0
  for (i in 1:nrow(x)) {
    dsum <- dsum + haversine(c(y[1], y[2]), c(x[i, 1], x[i, 2]))
  }
  return(dsum * km)
}

minimax <- function(y, x) {
  dsum <- 0
  for (i in 1:nrow(x)) {
    dsum <- max(dsum,haversine(c(y[1], y[2]), c(x[i, 1], x[i, 2])))
  }
  return(dsum * km)
}

distsFromCenter <- function(y, x) {
  di <- rep(0, 5)
  for (i in 1:5) {
    di[i] <- haversine(c(y[1], y[2]),
                       c(x[i, 1], x[i, 2]))
  }
  di <- as.matrix(di * km)
  row.names(di) <- row.names(x)
  colnames(di) <- "distance"
  return(di)
}
```

A.2 weberIter.R

```r
source("convert.R")
R <- 6371
km <- 0.6213712

weberIter <- function(x, itmax = 1000, eps = 1e-10, verbose = TRUE) {
  edist <- function(x, y) {
    return(sqrt(sum((x - y) ^ 2)))
  }
  n <- nrow(x)
  zold <- c(0, 0, 0)
  y <- matrix(0, n, 3)
  d <- rep(0, n)
  for (i in 1:nrow(x)) {
    y[i, ] <- convertSphericalToCartesian(x[i, ])
    zold <- zold + y[i, ]
  }
  zold <- zold / sqrt(sum(zold ^ 2))
  fold <- 0.0
  for (i in 1:nrow(x)) {
    d[i] <- 2 * R * asin(edist(R * zold, y[i, ]) / (2 * R))
    fold <- fold + 2 * R * asin(edist(R * zold, y[i, ]) / (2 * R))
  }
  itel <- 1
  repeat {
    znew <- c(0, 0, 0)
    fnew <- 0.0
    for (i in 1:n) {
      dd <- edist(zold, y[i, ] / R)
      ee <- dd * sqrt((1 - (dd / 2) ^ 2))
      znew <- znew + (1 / ee) * y[i, ] / R
    }
    znew <- znew / sqrt(sum(znew ^ 2))
    for (i in 1:nrow(x)) {
      d[i] <- 2 * R * asin(edist(R * znew, y[i, ]) / (2 * R))
      fnew <- fnew + d[i]
    }
    if (verbose == TRUE) {
      cat("itel ", formatC(itel, format = "d"),
          "fold ", formatC(fold, digits = 10, format = "f"),
          "fnew ", formatC(fnew, digits = 10, format = "f"),
          "weber point", formatC(znew, digits = 10, format = "f"),
          "\n")
    }
    if ((itel == itmax) || (max(abs(zold - znew)) < eps)) {
      break
    }
    zold <- znew
    fold <- fnew
    itel <- itel + 1
  }
  return(list(z = convertCartesianToSpherical(znew), f = km * fnew, d = km * d, itel =
}
```

A.3 utility.R

```
mPrint <- function(x,
                   digits = 10,
                   width = 15,
                   format = "f",
                   flag = "+") {
  print(noquote(
    formatC(
      x,
      digits = digits,
      width = width,
      format = format,
      flag = flag
    )
  ))
}
```

A.4 convert.R

```
convertSphericalToCartesian <- function(x, R = 6371) {
  x <- as.vector(convertDegToRad(x))
  y <- R * cos(x[1]) * cos(x[2])
  y <- c(y, R * cos(x[1]) * sin(x[2]))
  y <- c(y, R * sin(x[1]))
  return(y)
}

convertCartesianToSpherical <- function(x) {
  R <- sqrt(sum(x ^ 2))
  y <- asin(x[3] / R)
  y <- c(y, atan2(x[2], x[1]))
```

```r
  return(convertRadToDeg(y))
}

convertDegToRad <- function(x) {
  return(x * pi / 180)
}

convertRadToDeg <- function(x) {
  return(x * 180 / pi)
}
```

A.5 gcDistances.R

```r
gcDistancesHaversine <- function(x) {
  n <- nrow(x)
  dkm <- matrix(0, n, n)
  for (i in 1:n) {
    for (j in 1:n) {
      dkm[i, j] <- dkm[j, i] <-
        pracma::haversine(c(x[i, 1], x[i, 2]),
                          c(x[j, 1], x[j, 2]))
    }
  }
  row.names(dkm) <- row.names(x)
  colnames(dkm) <- row.names(x)
  return(dkm * km)
}

gcDistancesAsin <- function(x) {
  n <- nrow(x)
  dkm <- matrix(0, n, n)
  for (i in 1:n) {
    for (j in 1:n) {
      xx <- convertSphericalToCartesian(x[i, ])
      yy <- convertSphericalToCartesian(x[j, ])
      dd <- sqrt(sum((xx - yy) ^ 2))
      dkm[i, j] <- 2 * R * asin(dd / (2 * R))
    }
  }
  row.names(dkm) <- row.names(x)
  colnames(dkm) <- row.names(x)
  return(dkm * km)
}
```

A.6 haversine.R

```r
dyn.load("haversine.so")

gcDistance <- function(x, y) {
  h <- .C(
    "haversine",
    latHome = x[1],
    lonHome = x[2],
    latDest = y[1],
    lonDest = y[2],
    distance = 0.
  )
  return(h$distance)
}
```

A.7 haversine.c

```c


#include <math.h>

#define R 6371
#define TO_RAD (3.1415926536 / 180)

void haversine(double *th1, double *ph1, double *th2, double *ph2, double *hdist)
{
  double dx, dy, dz;
  double TH1 = *th1, PH1 = *ph1, TH2 = *th2, PH2 = *ph2;
  PH1 -= PH2;
  PH1 *= TO_RAD, TH1 *= TO_RAD, TH2 *= TO_RAD;
  dz = sin(TH1) - sin(TH2);
  dx = cos(PH1) * cos(TH1) - cos(TH2);
  dy = sin(PH1) * cos(TH1);
  *hdist = asin(sqrt(dx * dx + dy * dy + dz * dz) / 2) * 2 * R;
  return;
}
```

References

Borchers, H. W. 2023. *pracma: Practical Numerical Math Functions*. https://CRAN.R-project.org/package=pracma.

Katz, I. N., and L. Cooper. 1980. “Optimal Location on a Sphere.” *Computers & Mathematics with Applications* 6: 175–96.

R Core Team. 2024. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Weiszfeld, E. 1937. “Sur le Point par lequel la Somme des Distances de n Points Donnés est Minimum.” *Tohoku Mathematics Journal* 43: 355–86.

Weiszfeld, E., and F. Plastria. 2009. “On the Point for Which the Sum of the Distances to n Given Points Is Minimum.” *Annals of Operations Research* 167: 7–41.